13. ročník
Žilina 1. – 3. 7. 2025
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.fri.uniza.sk
OSSConf
OSSConf 2025: 1–7

O jedné mapě na zakázku, část 3: Obarvovací hra

PAVEL STŘÍŽ (CZ)

Abstrakt. Článek představuje variantu obarvovací hry v mapě/grafu. Barvy reprezentují cifry a ty se sčítají na hranicích a společných bodech. Idea pochází z teorému čtyř barev a generování zadání je principiálně stejné jako u sudoku. V pozadí se nejvíc užívá Python a Picat. V článku čtenář najde ukázky takové hry, ta není pro osm a víc barev tak jednoduchá, jak by se mohlo zdát. A hra by byla ještě těžší, kdyby se užila fiktivní mapa s průsečíky s mnoha oblastmi. Většina reálných map má hranici složenou jen ze dvou či tří oblastí. Je to výzva to zkusit.

Klíčová slova. Obarvovačka, Picat, Python, lineární programování.

ABOUT PREPARATION OF ONE MAP, PART 3: THE COLOURING GAME

Abstract. The article introduces a simple colouring game with graphs/maps using figures/sums in the areas and at the boundaries. The figures represent colours. The idea came from the Four Colour Theorem and the generation of the game is similar to the sudoku game generation. Python and Picat are the key programming languages. Samples of the game are included in the paper, it's not so easy to play when dealing with 8+ colours. It might be even harder when artificial graphs/maps would be used. Because most of the real-world maps have common points consisting from either two or three regions only. It's a challenge!

Keywords. Colouring game, Picat, Python, ILP, BLP.

1. Úvaha na odlehčení prvních dvou částí

Za sazbu by bylo poutavé mít hranici sousedů jinou barvou, např. součet barev oněch sousedů. Při experimentech jsem uvažoval tak, že když mám jeden kraj např. červenou barvou a druhý modrou, že jejich společná hranice by mohla být červená+modrá barva, tedy fialová. Ukázalo se to však jako nepraktické.

V teorii grafu lze mít hodnoty na uzlech, hranách či uvnitř oblastí. Barvy jsem měl uložené v poli, které má své číslování od nuly. Možná by se z toho dala udělat výpočetní hra typu sudoku. Tedy z grafu se všemi hodnotami a nějakými vztahy čísla odebírat, dokud zbylé vztahy tvoří jednoznačné řešení. Byla to myšlenka pro badatele. Naznačím zadání a řešení takové hry. Číslo v kraji reprezentuje jednu ze čtyř barev po obarvení. Sousedi nesmí mít stejnou barvu. Každý kraj má jen jednu barvu. Na hranicích dvou či tří krajů jsou příslušné součty reprezentující

míchání barev. Jako každá logická hra, i zde je jen jedno řešení. Pro vyšší počet barev jsem přidal podmínku, že každá barva musí být zastoupena alespoň jednou.

Pro potěšení hráčské vášně představujeme několik zadání vedoucí ke stejnému řešení. Vzkaz pro hackery: Pokud navrstvíte zadání na sebe, barvy téměř všech krajů se ukáží samy.

Jsou to ukázky na vyzkoušení hratelnosti. Asi by se dalo experimentovat s různým počtem uzlů, vztahů a počtem barev. Zde představuji vzorky pro 4, 8, 14 a plný počet 22 barev.

V druhé části tohoto pentaptychu jsme se zaměřili na výpočet počtu obarvení. To se nám zde hodí, protože je to nejúspornější cesta, jak mít nějaké konkrétní obarvení uložené. Ale to platí i pro bridž, šachy apod.

2. Pozadí generování

Jádro generování bylo již naznačeno. Přes Python se generuje zdrojový kód pro program Picat. Jen si musíme dát pozor, že SAT a CP rychle nedohledají model bez řešení. Také jejich řešení jsou systematicky uspořádána, pokud chceme řešení více náhodné, je potřeba v Picatu užít SMT či MIP. Obrácený postup, tedy do prázdné mapy přidávat vztahy dokud se nenajde model s jedním řešením, to jsem neužíval. Ale díky včasnému ukončení Picatu již při 2 řešeních je to možné.

Novinka vůči ukázce je tato část, ta zajistí, že každá barva bude užita alespoň jednou.

```
kolikbarev=4
for b in range(1,kolikbarev+1):
  data=[]
  for o in okresy:
    data.append("Xo"+str(o)+"b"+str(b))
  kam.write("+".join(data)+" #>= 1,\n")
kam.write("\n")
```

Místo jedničky se můžeme snažit o rovnoměrné rozložení, třeba touto cestou:

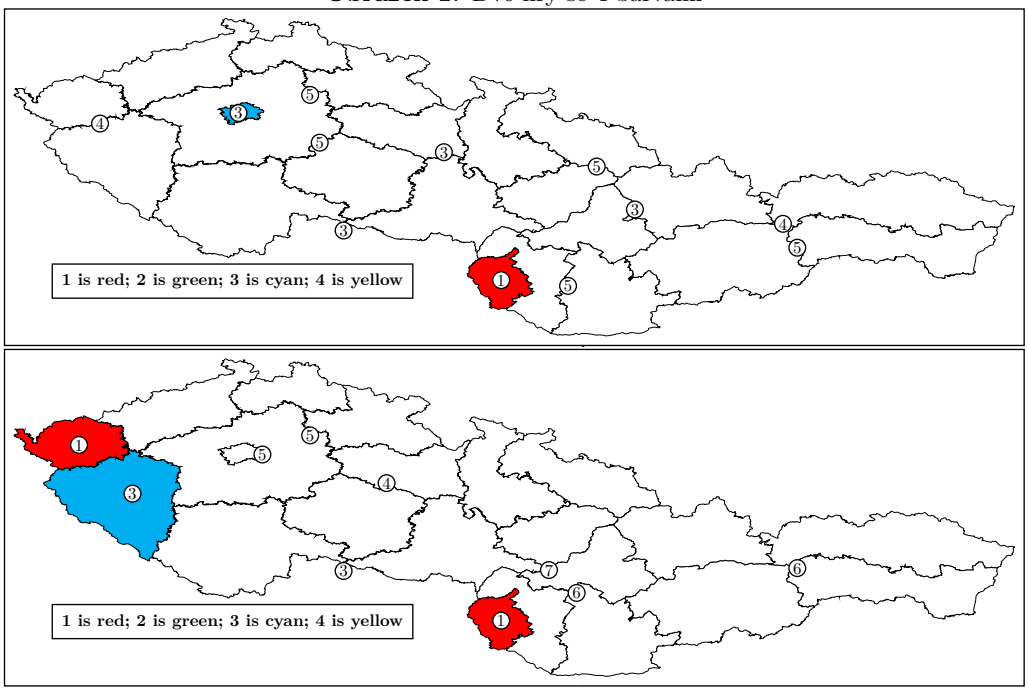
```
mohu=math.floor(len(okresy)/kolikbarev)
[...]
  kam.write("+".join(data)+" #>= "+str(mohu)+",\n")
[...]
```

Pro 22 krajů či okresů a 4 barvách to znamená, že každá barva bude užita alespoň pětkrát. U svých vzorků používám jedničku.

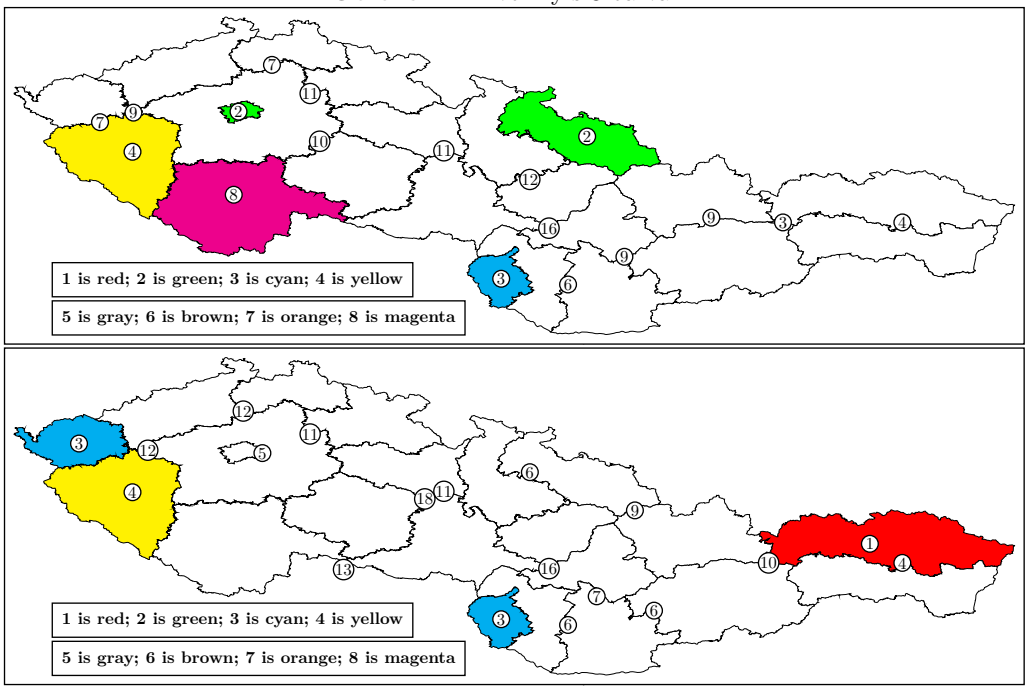
V předchozí ukázce jsme si generovali jedno řešení. To se užije na přípravu zadání hry. Touto cestou lze omezit počet řešení na dvě. To znamená, že pokud se najde víc řešení než jedno, nelze to považovat za logickou hru.

```
kam.write("L=findall(Vypis,solve([$limit(2)],Vypis)),\n")
kam.write("writeln(L).\n")
```

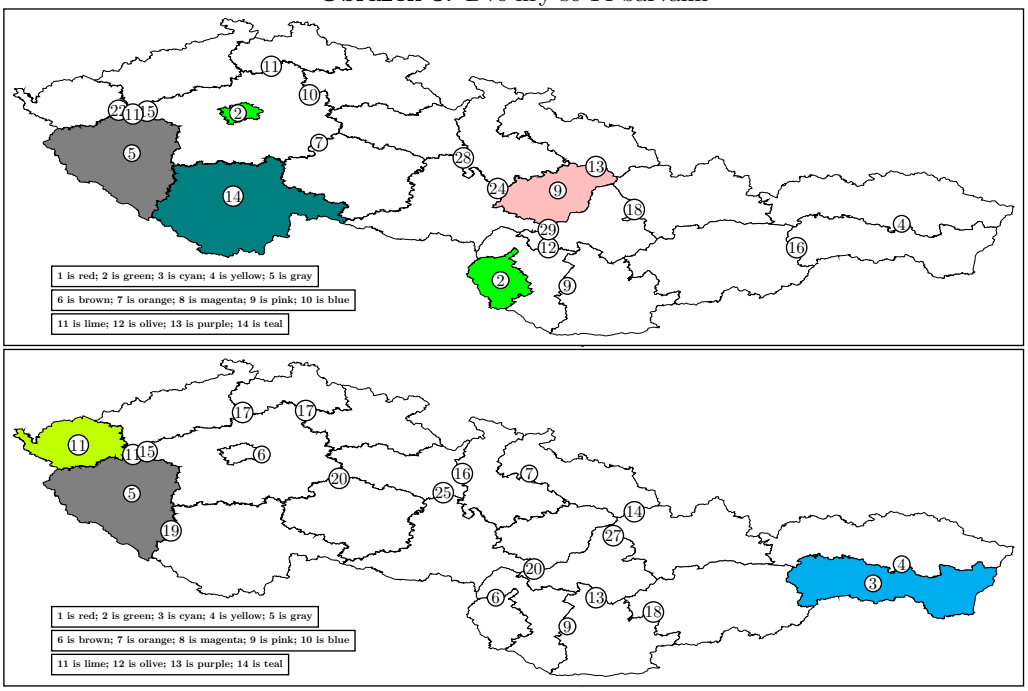
Obrázek 1. Dvě hry se 4 barvami



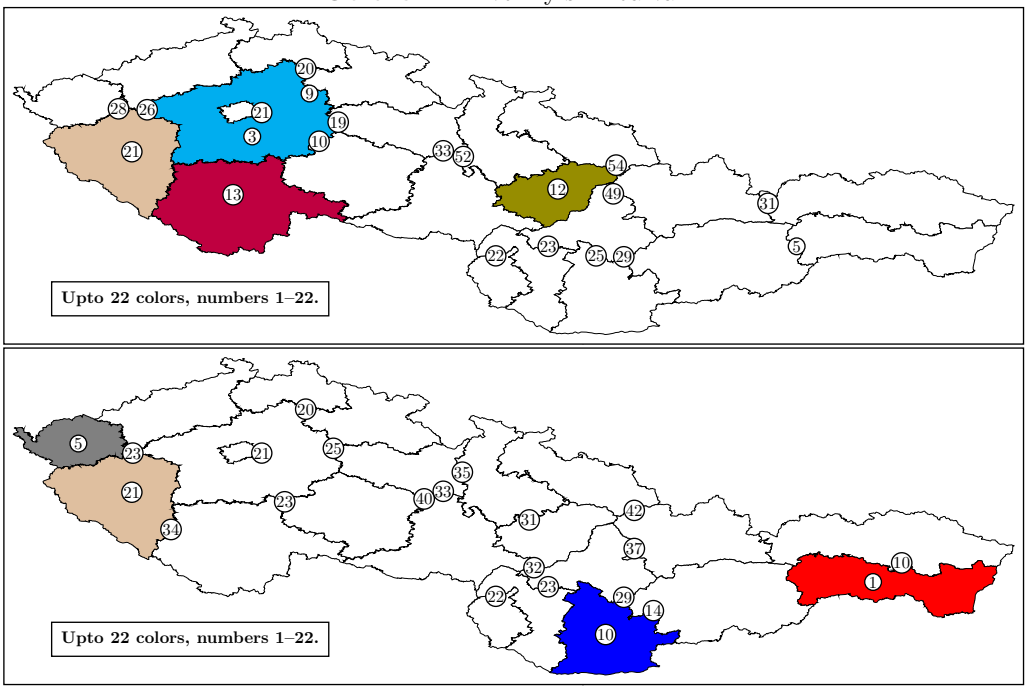
Obrázek 2. Dvě hry s 8 barvami



Obrázek 3. Dvě hry se 14 barvami



Obrázek 4. Dvě hry s 22 barvami



Nu, a jsme u přípravy zadání hry z řešení. Podobně to realizuji u sudoku. Vezmeme všechny vztahy, nyní jich je 87:

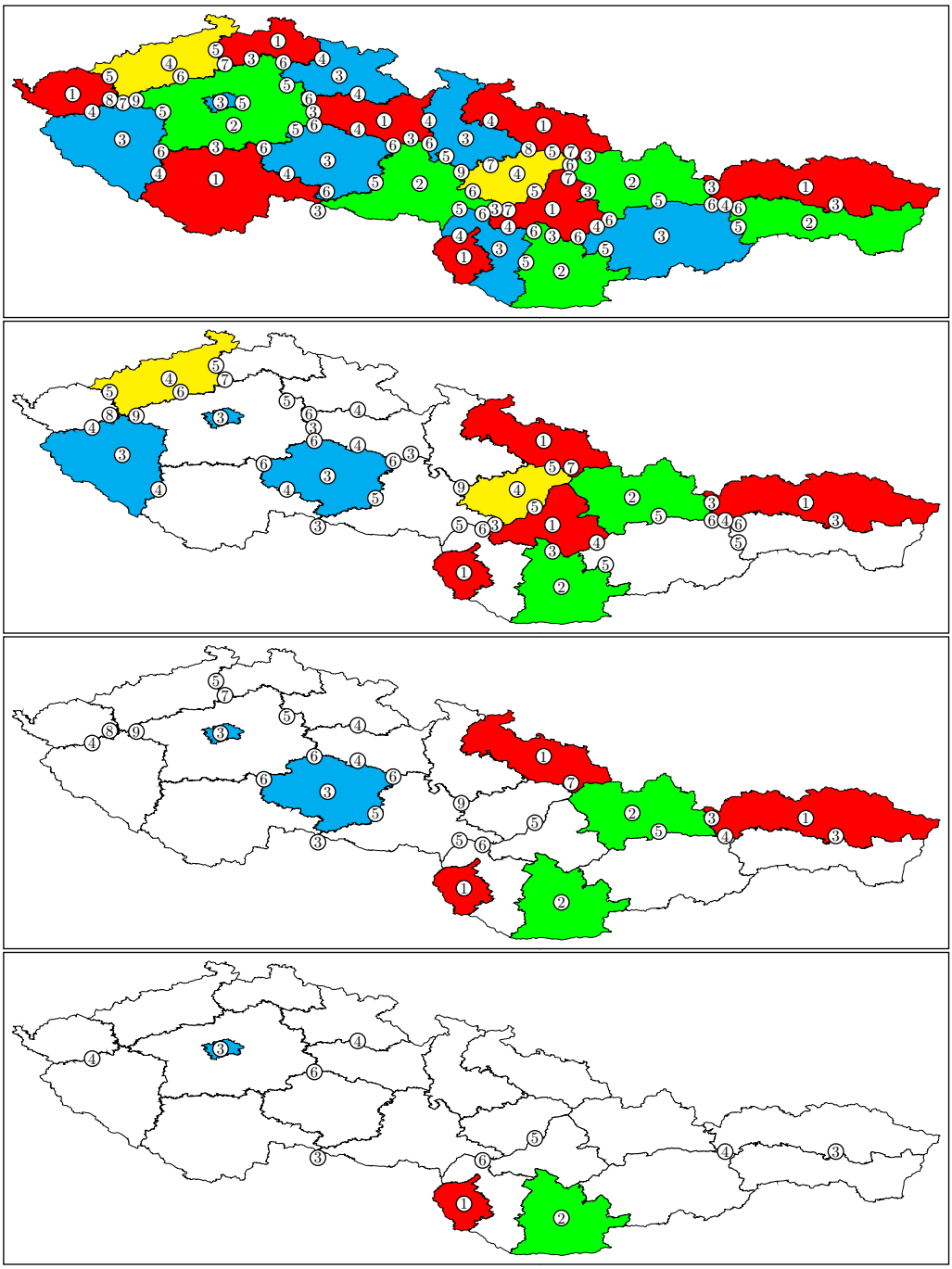
- 14+8 krajů za ČR a SR,
- 43 hranic dvou krajů, a,
- 22 míst, kde se spojují tři kraje (víc než tři pro tuto mapu neexistuje).

Podmínky byly jednoduché. Buď musí být určitá barva v kraji, nebo součet na hranici jisté číslo, nebo součet tří čísel reprezentující barvu na spojnici tří krajů.

Zde jsou surová data, kdyby si někdo chtěl experiment tohoto typu zkusit na vlastní kůži.

```
kraje={
   0: {"name":"Hlavní město Praha",   "sousedi":[1]},
   1: {"name":"Středočeský kraj",     "sousedi":[0,2,3,5,6,7,8,9]},
   2: {"name":"Jihočeský kraj",       "sousedi":[1,3,9,10]},
   3: {"name":"Plzeňský kraj",        "sousedi":[1,2,4,5]},
   4: {"name":"Karlovarský kraj",     "sousedi":[3,5]},
   5: {"name":"Ústecký kraj",         "sousedi":[1,3,4,6]},
   6: {"name":"Liberecký kraj",       "sousedi":[1,5,7]},
   7: {"name":"Královéhradecký kraj", "sousedi":[1,6,8]},
   8: {"name":"Pardubický kraj",      "sousedi":[1,7,9,10,11]},
   9: {"name":"Kraj Vysočina",        "sousedi":[1,2,8,10]},
  10: {"name":"Jihomoravský kraj",    "sousedi":[2,8,9,11,13,15,16]},
  11: {"name":"Olomoucký kraj",       "sousedi":[8,10,12,13]},
  12: {"name":"Moravskoslezský kraj", "sousedi":[11,13,18]},
  13: {"name":"Zlínský kraj",         "sousedi":[10,11,12,16,18]},
  14: {"name":"Bratislavský kraj",    "sousedi":[15]},
  15: {"name":"Trnavský kraj",        "sousedi":[10,14,16,17]},
  16: {"name":"Trenčínský kraj",      "sousedi":[10,13,15,17,18,19]},
  17: {"name":"Nitranský kraj",       "sousedi":[15,16,19]},
  18: {"name":"Žilinský kraj",        "sousedi":[12,13,16,19,20]},
  19: {"name":"Banskobystrický kraj", "sousedi":[16,17,18,20,21]},
  20: {"name":"Prešovský kraj",       "sousedi":[18,19,21]},
  21: {"name":"Košický kraj",         "sousedi":[19,20]},
  }
dvojice=[ [0,1], [1,2], [1,3], [1,5], [1,6], [1,7], [1,8], [1,9], [2,3], [2,9],
  [2,10], [3,4], [3,5], [4,5], [5,6], [6,7], [7,8], [8,9], [8,10], [8,11],
  [9,10], [10,11], [10,13], [10,15], [10,16], [11,12], [11,13], [12,13],
  [12,18], [13,16], [13,18], [14,15], [15,16], [15,17], [16,17], [16,18],
  [16,19], [17,19], [18,19], [18,20], [19,20], [19,21], [20,21] ]
trojice=[ [1,8,9], [1,7,8], [1,6,7], [1,5,6], [1,3,5], [1,2,3], [1,2,9],
  [2,9,10], [3,4,5], [8,9,10], [8,10,11], [10,15,16], [10,13,16], [10,11,13],
  [11,12,13], [12,13,18], [13,16,18], [15,16,17], [16,17,19], [16,18,19],
  [18,19,20], [19,20,21] ]
```

Nechám si je zamíchat a pak se postupně jeden po druhém snažím odebrat. Pokud Picat po odebrání najde jen jedno řešení, znamená to, že takový vztah lze z ILP modelu odebrat. Pokud najde 2 řešení (technicky jich může být mnohem víc, ale to nás netíží), tak takovou podmínku odebrat nelze, musí v modelu zůstat.



Pokud chceme jednodušší či středně těžkou variantu, ukončíme prohledávání dříve než po testu všech 87 podmínek. Pokud chceme co nejtěžší variantu, testujeme co nejvíc vztahů.

Výpis z cyklu vypadá zhruba takto:

```
Generuji soubor pro Picat, položka č. 1/87...: 1
Generuji soubor pro Picat, položka č. 2/87...: 1
[...]
Generuji soubor pro Picat, položka č. 49/87...: 1
Generuji soubor pro Picat, položka č. 50/87...: 2
Generuji soubor pro Picat, položka č. 51/87...: 1
[...]
Generuji soubor pro Picat, položka č. 86/87...: 2
Generuji soubor pro Picat, položka č. 87/87...: 2
```

Kde Picat vrátil jedno řešení, podmínku z ILP modelu jsem odebral, pokud 2, tak jsem testovanou podmínku ponechal. Na předchozí straně je grafický náhled postupného odebírání podmínek: na začátku, po 40, 60 a testu všech 87 podmínek. `X` značí, že se jedná o neznámou, `o` značí oblast, u původních testů okres.

Po testu všech podmínek zůstalo v ILP modelu následující. Z posledního obrázku si rychle přiřadíte. Sazba byla jednoduchá: vypiš součty a vyplň kraj barvou jen v případě, je-li vztah součástí ILP modelu.

```
Xo14           #= 1,   % Bratislavský kraj.
Xo0            #= 3,   % Hlavní město Praha.
Xo2 +Xo10      #= 3,   % Jihočeský + Jihomoravský kraj.
Xo16+Xo10+Xo15 #= 6,   % Trenčínský + Jihomoravský + Trnavský kraj.
Xo17           #= 2,   % Nitranský kraj.
Xo20+Xo21      #= 3,   % Prešovský + Košický kraj.
Xo8 +Xo7       #= 4,   % Pardubický + Královéhradecký kraj.
Xo16+Xo13      #= 5,   % Trenčínský + Zlínský kraj.
Xo19+Xo20      #= 4,   % Banskobystrický + Prešovský kraj.
Xo8 +Xo1 +Xo9  #= 6,   % Pardubický + Středočeský + Kraj Vysočina.
```

3. Závěrem

Pokud se alespoň trochu výrobně zadaří, mapa ČR a SR by měla tvořit výstavu na OSSConf a poté trvalejší vyvěšení by mělo být někde v budově Žilinské univerzity, na Fakultě řízení a informatiky. Vzorky z této hry do velké mapy zařadím též, někam, pokud a bude-li volné místo.

Kontaktní adresa

Ing. Pavel Stříž, Ph.D., Nakladatelství Martin Stříž, U Škol 940, Bučovice, okres Vyškov, 685 01, Česká republika,

E-mailová adresa: pavel@striz.cz